One-Class Genetic Programming

Robert Curry and Malcolm I. Heywood

Dalhousie University
6050 University Avenue
Halifax, NS, Canada
B3H 1W5
{rcurry,mheywood}@cs.dal.ca

Abstract. One-class classification naturally only provides one-class of exemplars, the target class, from which to construct the classification model. The one-class approach is constructed from artificial data combined with the known in-class exemplars. A multi-objective fitness function in combination with a local membership function is then used to encourage a co-operative coevolutionary decomposition of the original problem under a novelty detection model of classification. Learners are therefore associated with different subsets of the target class data and encouraged to tradeoff detection versus false positive performance; where this is equivalent to assessing the misclassification of artificial exemplars versus detection of subsets of the target class. Finally, the architecture makes extensive use of active learning to reinforce the scalability of the overall approach.

Keywords: One-Class Classification, Coevolution, Active Learning, Problem Decomposition.

1 Introduction

The ability to learn from a single class of exemplars is of importance under domains where it is not feasible to collect exemplars representative of all scenarios e.g., fault or intrusion detection; or possibly when it is desirable to encourage fail safe behaviors in the resulting classifiers. As such, the problem of one-class learning or 'novelty detection' presents a unique set of requirements from that typically encountered in the classification domain. For example, the discriminatory models of classification most generally employed might formulate the credit assignment goal in terms of maximizing the separation between the in- and out-class exemplars. Clearly this is not feasible under the one-class scenario. Moreover, the one-class case often places more emphasis on requiring fail safe behaviors that explicitly identify when data differs from the target class or 'novelty detection'.

Machine learning algorithms employed under the one-class domain therefore need to address the discrimination/ novelty detection problem directly. Specific earlier works include Support Vector Machines (SVM) [1,2,3], bottleneck neural

L. Vanneschi et al. (Eds.): EuroGP 2009, LNCS 5481, pp. 1–12, 2009.

networks [4] and a recent coevolutionary genetic algorithm approach based on an artificial immune system [5] (for a wider survey see [6,7]).

In particular the one-class SVM model of Schölkopf relies on the correct identification of "relaxation parameters" to separate exemplars from the origin (representing the second unseen class) [1]. Unfortunately, the values for such parameters vary as a function of the data set. However, a recent work proposed a kernel autoassociator for one-class classification [3]. In this case the kernel feature space is used to provide the required non-linear encoding, this time in a very high dimensional space (as opposed to the MLP approach to the encoding problem). A linear mapping is then performed to reconstruct the original attributes as the output. Finally, the work of Tax again uses a kernel based one-class classifier. This approach is distinct in that data is artificially generated to aid the identification of the most concise hypersphere describing the in-class data [2]. Such a framework builds on the original support vector data description model, whilst reducing the significance of specific parameter selections. The principle drawback, however, is that tens or even hundreds of thousands of artificially generated training exemplars are required to build a suitably accurate model [2]. The work proposed in this paper uses the artificial data generation model of Tax, but specifically addresses the training overhead by employing an active learning algorithm. Moreover, the Genetic Programming (GP) paradigm provides the opportunity to solve the problem using an explicitly multiple objective model, where this provides the basis for cooperative coevolutionary problem decomposition.

2 Methodology

The general principles of the one-class GP (OCGP) methodology, as originally suggested in [8], is comprised of four main components:

(1) **Local membership function:** Conventionally, GP programs provide a mapping between the multi-dimensional attribute space and a real-valued one-dimensional number line called the *gpOut* axis. A binary switching function (BSF), as popularized by Koza, is then used to map the one-dimensional 'raw' *gpOut* to one of two classes, as shown in Fig. 1(a) [9]. However, a BSF assumes that the two classes can be separated at the origin. Moreover, under a one-class model of learning – given that we have no information on the distribution of out-class data – exemplars belonging to the unseen classes are just as likely to appear on either side of the origin, resulting in high false positive rates. Therefore, instead of using the 'global' BSF, GP individuals utilize a Gaussian or 'local' membership function (LMF), Fig. 1(b). A small region of the *gpOut* axis is therefore evolved for expressing in-class behavior, where this region is associated with a subset of the target distribution encountered during training. In this way GP individuals act as novelty detectors, as any region on the *gpOut* axis other than that of the LMF is associated with the out-class conditions; thus supporting conservative generalization properties when deployed. Moreover, instead of a single classifier providing a single mapping for all in-class exemplars,

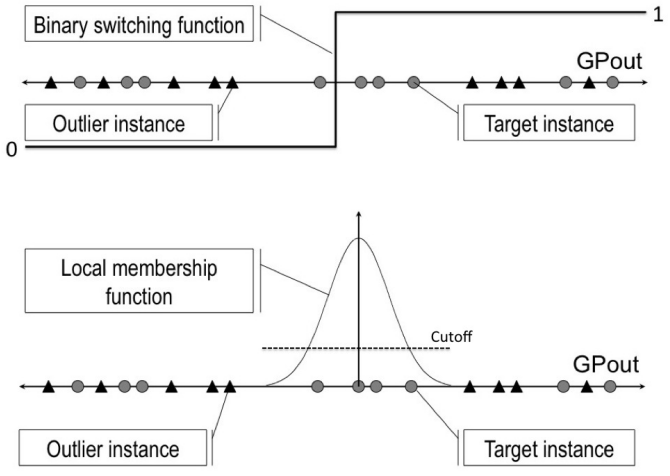


Fig. 1. (a) 'Global' binary switching function vs. (b) Gaussian 'local' membership function

our goal will be to encourage the coevolution of multiple GP novelty detectors to map unique subsets of the in-class exemplars to their respective LMF.

(2) **Artificial outlier generation:** In a conventional binary classification problem the decision boundaries between classes are supported by exemplars from each class. However, in a one-class scenario it is much more difficult to establish appropriate and concise decision boundaries since they are only supported by the target class. Therefore, the approach we adopt for developing one-class classifiers is to build the outlier class data artificially and train using a two class model. The 'trick' is to articulate this goal in terms of finding an optimal trade-off between detection and false positive rates as opposed to explicitly seeking solutions with 'zero error'.

(3) **Balanced block algorithm (BBA):** When generating artificial outlier data it is necessary to have a wider range of possible values than the target data and also to ensure that the target data is surrounded in all attribute directions. Therefore, the resulting 'two-class' training data set tends to be unbalanced and large; implying artificial data partitions in the order of tens of thousands of exemplars (as per Tax [2]). The increased size of the training data set has the potential to significantly increase the training overhead of GP. To address this overhead the BBA active learning algorithm is used [10]; thus fitness evaluation is conducted over a much smaller subset of training exemplars, dynamically identified under feedback from individuals in the population, Fig. 2.

(4) **Evolutionary multi-objective optimization (EMO):** EMO allows multiple objectives to be specified, thereby providing a more effective way to express the quality of GP programs. Moreover, EMO provides a means of comparing individuals under multiple objectives without resorting to *a priori* scalar weighting

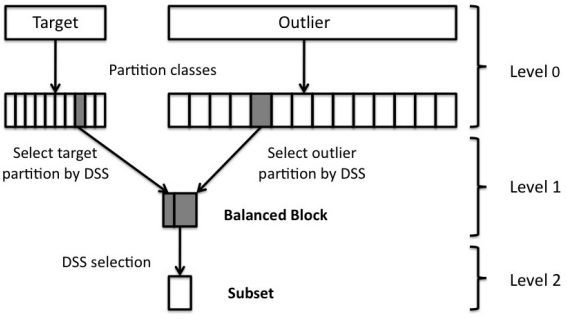


Fig. 2. The balanced block algorithm first partitions the target and outlier classes (Level 0) [10]. Balanced blocks of training data are then formed by selecting a partition from each class by dynamic subset selection (Level 1). For each block multiple subsets are then chosen for GP, training again performed by DSS (Level 2).

functions. In this way the overall OCGP classifier is identified through a cooperative approach that supports the simultaneous development of multiple programs from a *single* population.

The generation of artificial data in and around the target data means that outlier data lying within the actual target distribution cannot be avoided. Thus, when attempting to classify the training data it is necessary to cover as much of the target data as possible (i.e., maximize detection rate), while also minimizing the amount of outlier data covered (i.e., minimize false positive rate); the first two objectives. Furthermore, it is desirable to have an objective to encourage diversity among solutions by actively rewarding non-overlapping behavior between the coverage of different classifiers as evolved from the same population. Finally, the fourth objective encourages solution simplicity, thus reducing the likelihood of overfitting and promoting solution transparency.

GP programs are compared by their objectives using the notion of *dominance*, where a classifier A is said to *dominate* classifier B if it performs at least as well as B in all objectives and better than B in at least one objective. *Pareto ranking* then combines the objectives into a scalar fitness by assigning each classifier a *rank* based on the number of classifiers by which it is dominated [11,12]. A classifier is said to be *non-dominated* if it is not dominated by any other classifier in the population and has a rank of zero. The set of all non-dominated classifiers is referred to as the *Pareto front*. The Pareto front programs represent the current best trade-offs of the multi-objective criteria providing a range of candidate solutions. Pareto front programs influence OCGP by being favored for reproduction and update the archives of best programs which determine the final OCGP classifiers.

2.1 OCGP Algorithm

The general framework of our algorithm is described by the flowchart in Fig. 3. The first step is to initialize the random GP population of programs and the

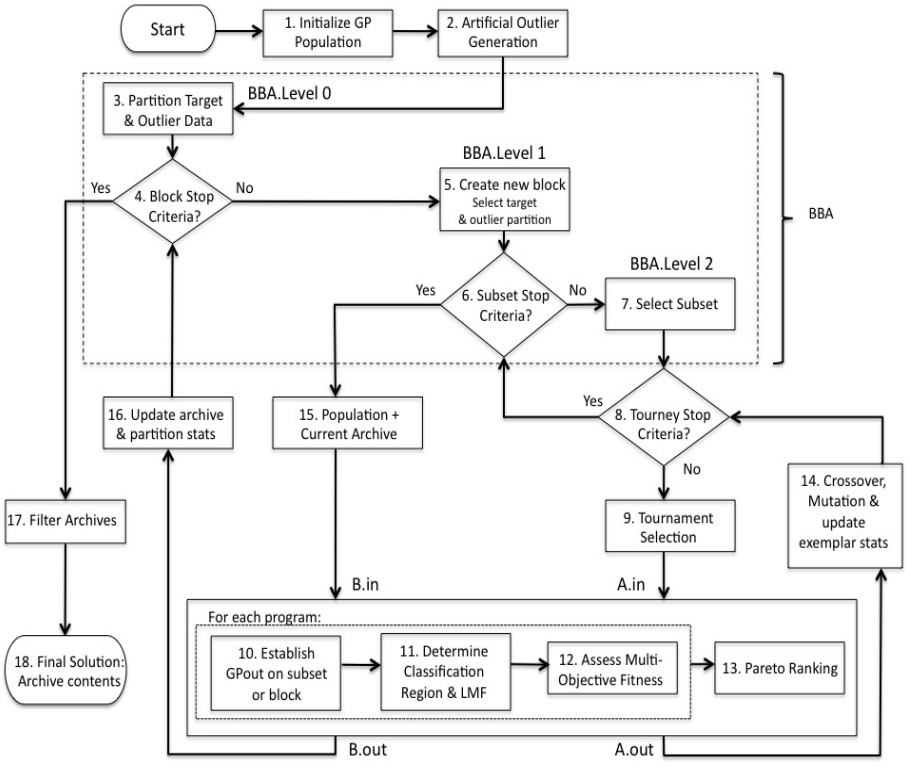


Fig. 3. Framework for OCGP assuming target data is provided as input

necessary data structures, Step 1. OCGP is not restricted to a specific form of GP but in this work a linear representation is assumed. Artificial outlier data is then generated in and around the provided target data, Step 2.

The next stage outlines the three levels of the balanced block algorithm (BBA), Fig. 2, within the dashed box. Level 0, Step 3, partitions the target and outlier data. The second level, Step 5, selects a target and outlier partition to create the Level 1 balanced block. At Level 2, Step 7, subsets are selected from the block to form the subset of exemplars over which fitness evaluation is performed (steady state tournament).

The next box outlines how individuals are evaluated. First programs are evaluated on the current data subset to establish the corresponding *gpOut* distribution, Step 10. The classification region is then determined to parameterize the LMF, Step 11, and the multi-objective fitness is established, Step 12. Once all programs have their multi-objective fitness the programs can be Pareto ranked, Step 13.

The Pareto ranking determines the tournament winners, or parents, from which genetic operators can be applied to create children, Step 14. In addition parent programs update the difficulty of exemplars in order to influence future subset selections. That is to say, previous performance (error) on artificial–target

class data is used to guide the number of training subsets sampled from level 1 blocks. As such, difficulty values are averaged across the data in level 1 and 2, Step 6 and 8 respectively.

Once training is complete at level 2, the population and the archives associated with the current target partitions are combined, Step 15, and evaluated on the Level 1 block (Step 10 through Step 13). The archive of the target partition is then updated with the resulting Pareto front, Step 16, and partition difficulties updated in order to influence future partition selections and the number of future subset iterations. The change in partition error rates for each class is also used to determine the block stop criteria, Step 4. Once the block stop criteria has been met the archives are filtered with any duplicates across the archives being removed, Step 17, and the final OCGP classifier consists of the remaining archive programs. More details of the BBA are available from [10].

2.2 Developments

Relative to the above, this work introduces the following developments:

Clustering. In the original formulation of OCGP the classification region for each program (i.e., LMF location) was determined by dividing a program's *gpOut* axis into class-consistent 'regions' (see Fig. 4) and the corresponding class separation distance (1), or *csd*, between adjacent regions estimated. The target region that maximizes *csd* with respect to the neighboring outlier regions has the best separability and is chosen as the classification region for the GP program (determining classification region at Fig. 3 Step 11).

$$csd_{\,0/1} = \frac{|\mu_0 - \mu_1|}{\sqrt{\sigma_0^2 + \sigma_1^2}} \tag{1}$$

In this work the LMF is associated with the most dense region of the *gpOut* axis i.e., independent of the label information. Any artificial exemplars lying within the LMF might either result in the cluster being penalized once the fitness criteria is applied or be considered as outliers generated by the artificial

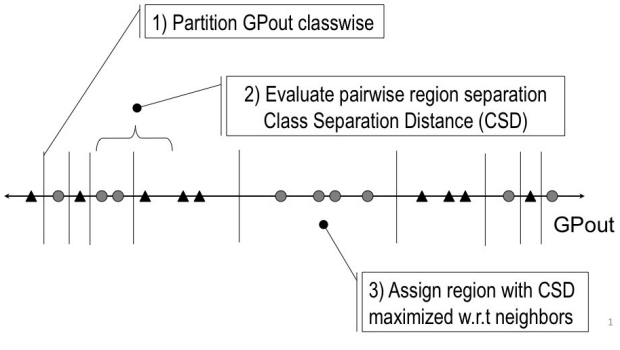


Fig. 4. Determining GP classification region by class separation distance

data generation model. In effect we establish a 'soft' model of clustering (may contain some artificial data points) in place of the previous 'hard' identification of clusters (no clusters permitted with any artificial data points).

To this end, subtractive clustering [13] is used to cluster the one-dimensional *gpOut* axis and has the benefit of not requiring *a priori* specification of the number of clusters. The mixed content of a cluster precludes the use of a class separation distance metric. Instead the sum squared error (SSE) of the exemplars within the LMF are employed. The current GP program's associated LMF returns confidence values for each of the exemplars in the classification region. Therefore, the error for each type of exemplar can be determined by subtracting their confidence value from their actual class label (i.e., reward for target exemplars in classification region and penalize for outliers). The OCGP algorithm using *gpOut* clustering will be referred to as OCGPC.

Caching. The use of the clustering algorithm caused the OCGPC algorithm to run much more slowly than OCGP. The source was identified to be the clustering of the entire GP population and the current archive on the entire Level 1 block (Fig. 3 Step 15). Therefore, instead of evaluating the entire population and archive on the much larger Level 1 block, the mean and standard deviation is cached from the subset evaluation step. Caching was introduced in both the OCGP and OCGPC algorithms and was found to speed up training time without negatively impacting classification performance.

Overlap. The overlap objective has been updated from the previous work to compare tournament programs against the current archive instead of comparing against the other tournament programs (assessing multi-objective fitness at Step 12). Individuals losing a tournament are destined to be replaced by the search operators, thus should not contribute to the overlap evaluation. Moreover, comparison against the archive programs is more relevant, as they represent the current best solution to the current target partition (i.e., target exemplars already covered) and thus encourages tournament programs to classify the remaining uncovered target exemplars.

Artificial outlier generation. Modifications have been made in order to improve the quality of the outlier data (Fig. 3 Step 2). Previously a single radius, R, was determined by the attribute of the target data having the largest range and was then used as the radius for all attribute dimensions when creating outliers. If a large disparity exists between attribute ranges, this can lead to large volumes of the outlier distribution with little to no relevance to the target data. Alternatively, a vector $\overline{R}$ of radii is used consisting of a radius for each attribute. Additionally, when the target data consists of only non-negative values, negative outlier attribute values are restricted to within a close proximity to zero.

3 Experiments

In contrast to the previous performance evaluation [8], we concentrate on benchmarking against data sets that have large unbalanced distributions in the

Table 1. Binary classification datasets. The larger in-class partition of Adult, Census and Letter-vowel was matched with a larger artificial exemplar training partition.

Dataset	**Adult**		**Census**		**Letter-vowel**	
Features	14		40		16	
Class	Train	Test	Train	Test	Train	Test
0	50,000	11,355	50,000	34,947	50,000	4,031
1	7,506	3,700	5,472	2,683	2,660	969
Total	57,506	15,055	55,472	37,630	52,660	5,000
Dataset	**Letter-a**		**Letter-e**		**Mushroom**	
Features	16		16		22	
Class	Train	Test	Train	Test	Train	Test
0	10,000	4,803	10,000	4,808	10,000	872
1	574	197	545	192	1,617	539
Total	10,574	5,000	10,545	5,000	11,617	1,411

underlying exemplar distribution, thus are known to be difficult to classify under binary classification methods. Specifically, the Adult, Census-Income (KDD), Mushroom and Letter Recognition data sets from the UCI machine learning repository [14] were utilized (Table 1). The Letter Recognition data set was used to create three one-class classification data sets where the target data was alternately all vowels (Letter-vowel), the letter 'a' (Letter-a) and the letter 'e' (Letter-e). For the Adult and Census data sets a predefined training and test partition exists. For the Letter Recognition and Mushroom data sets the data was first divided into a 75% training and 25% test data set while maintaining the class distributions of the original data. The class 0 exemplars were removed from the training data set to form the one-class target data set.

Training was performed on a dual G4 1.33 GHz Mac Server with 2 GB of RAM. All experiments are based on 50 GP runs where runs differ only in their choice of random seeds for initializing the population while all other parameters remain unchanged. Table 2 lists the common parameter settings for all runs.

The OCGP algorithm results are compared to results found by a one-class support vector machine (OC ν-SVM) [1] and a one-class or bottleneck neural network (BNN)[1], where both algorithms are trained on the target data alone. Additionally, the OCGP results will be compared to a two-class support vector machine (ν-SVM) which use both the artificial outlier and the target data. The two-class SVM is used as a baseline binary classification algorithm in order to assess to what degree the OCGP algorithms are able to provide a better characterization of the problem i.e., both algorithms are trained on the target and artificial outlier data. Comparison against Tax's SVM was not possible as the current implementation does not scale to large data sets.

[1] Unlike the original reference ([4]) the BNN was trained using the efficient second Conjugate Gradient weight update rule with tansig activation functions; both of which make a significant improvement over first order error back-propagation.

Table 2. Parameter Settings

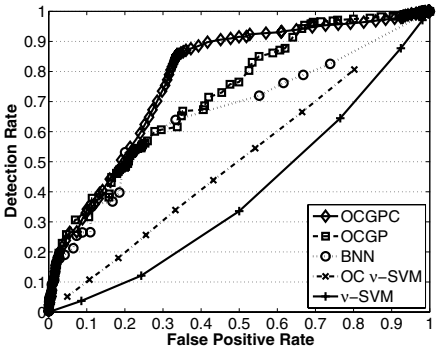
Dynamic Page-Based Linear GP			
Population size	125	Tournament Size	4
Max # of pages	32	Number of registers	8
Page size	8 instructions	Instr. prob. 1, 2 or 3	0/5, 4/5, 1/5
Max page size	8 instructions	Function set	$\{+, -, \times, \div\}$
Prob. Xover, Mut., Swap	0.9, 0.5, 0.9	Terminal set	{# of attributes}
Balanced Block Algorithm Parameters			
Target Partition Size	$\approx \frac{\#Patterns}{\#Archives}$	Max subset iterations	10
Outlier Partition Size	500	Tourneys per subset	6
Max block selections	2000	Level 2 subset size	100
Archive Parameters			
Number of Archives	Adult = 15, Census = 11, Letter-vowel = 10, Letter-a = 6, Letter-e = 6, Mushroom = 4		
Archive Size	10		

The algorithms are compared in terms of ROC curves of (FPR, DR) pairs on test data sets (Fig. 5). Due to the large number of runs of the OCGP algorithms and the multiple levels of voting possible, plotting all of the OCGP solutions becomes too convoluted. Alternatively, only non-dominated solutions will be shown, where these represent the best-case OCGP (FPR, DR) pairs over the 50 runs. Similarly only the non-dominated solutions of the bottle-neck neural networks will be shown, while for the other algorithms only a small number of solutions are present and so all results will be plotted. Comments will be made as follows on an algorithm-by-algorithm basis:

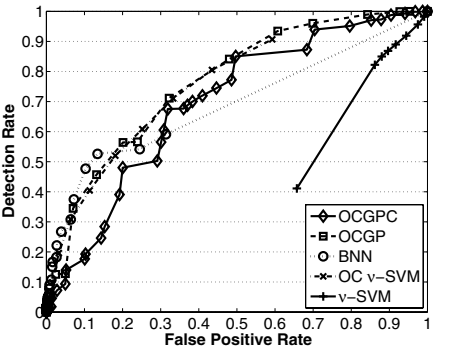
OCGPC. Of the two one-class GP models the cluster variant for establishing the region of interest on the *gpOut* axis described in Sect. 2.2 appeared to be the most consistent. Specifically, OCGPC provided the best performing curves under the Adult and Vowel data sets (Fig. 5(a) and (c)) and consistently the runner up under all but the Census data set. In each of the three cases where it appears as a runner up it was second to the BNN. However, GP retains the advantage of indexing a subset of the attributes; whereas multi-layer neural networks index all features – a bias of NN methods in general and the autoassociator method of deployment in particular.

OCGP. When assuming the original region based partitioning of *gpOut*, ROC performance decreases significantly with strongest performance on the Census data set; and runner up performance under Adult and Mushroom. Performance on the remaining data sets might match or better the one-class ν-SVM, but generally worse than BNN or OCGPC.

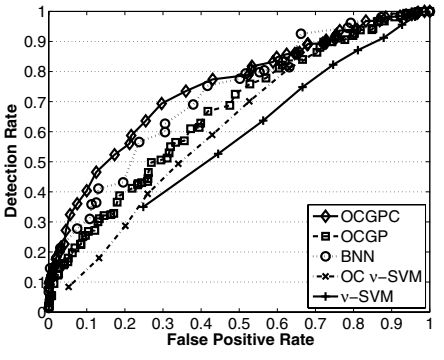
BNN. As remarked above the BNN was the strongest one-class model, with best ROC curves on Letter 'a', 'e', and mushroom; and joint best on Census (with OCGP). Moreover, the approach was always better than the SVM methods benchmarked in this work.



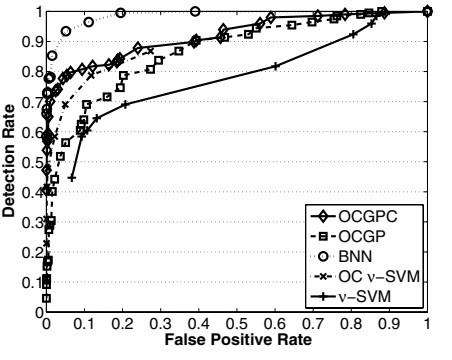
(a) Adult



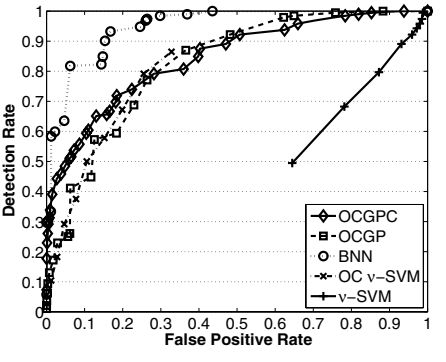
(b) Census



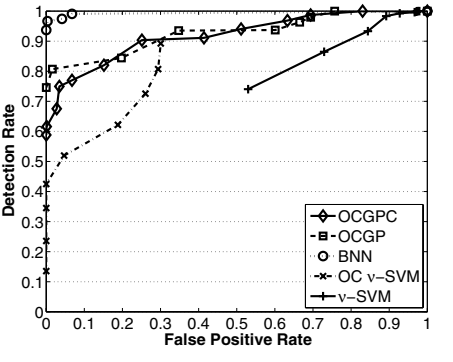
(c) Vowel



(d) Letter 'a'



(e) Letter 'e'



(f) Mushroom

Fig. 5. Test dataset ROC curves

SVM methods. Neither SVM method – one-class or binary SVM trained on the artificial versus target class – performed well. Indeed the binary SVM was at best degenerate on all but the Letter 'a' data set; resulting in it being ranked worst in all cases. This indicates that merely combining artificial data with target class data is certainly not sufficient for constructing one-class learners as the learning objective is not correctly expressed. The one-class SVM model generally avoided degenerate solutions, but never performed better than OCGPC or BNN.

4 Conclusion

A Genetic Programming (GP) classifier has been proposed in this work for the one-class classification domain. Four main components of the algorithm have been identified. Artificial outlier generation is used to establish more concise boundaries and to enable the use of a two-class classifier. The active learning algorithm BBA tackles the class imbalance and GP scalability issues introduced by the large number of artificial outliers required. Gaussian 'local' membership functions allow GP programs to respond to specific regions of the target distribution and act as novelty detectors. Evolutionary multi-objective optimization drives the search for improved target regions, while allowing for the simultaneous development of multiple programs that cooperate towards the overall problem decomposition through the objective for minimizing overlapping coverage.

A second version of the OCGP algorithm is introduced, namely OCGPC, which determines classification regions by clustering the one-dimensional *gpOut* axis. In addition, 'caching' of the classification regions is introduced to both algorithms, in order to eliminate the need to redetermine classification regions over training blocks. Caching reduces training times without negatively impacting classification performance. Modifications were also made to improve the quality of generated artificial outliers and to the objectives used to determine classification regions, including the use of the sum-squared error and improving the overlap objective by comparing to only the current best archive solutions.

The OCGP and OCGPC algorithms were evaluated on six data sets larger than previously examined. The results were compared against two one-class classifiers trained on target data alone, namely one-class ν-SVM and a bottleneck neural network (BNN). An additional comparison was made with a two-class SVM trained on target data and the generated artificial outlier data. The OCGPC and BNN models were the most consistent performers overall; thereafter model preference might be guided by the desirability for solution simplicity. In this case the OCGPC model makes additional contributions as it operates as a classifier as opposed to an autoassociator i.e., autoassociators are unable to simplify solutions in terms of attributes indexed. Future work will concentrate in this direction and to applying OCGPC to learning without artificial data.

Acknowledgements

BNN and SVM models were built in MATLAB and LIBSVM respectively. The authors acknowledge MITACS, NSERC, CFI and SwissCom Innovations.

References

1. Scholköpf, B., Platt, J., Shawe-Taylor, J., Smola, A., Williamson, R.: Estimating the support of a high-dimensional distribution. Neural Computation 13, 1443–1471 (2001)
2. Tax, D., Duin, R.: Uniform object generation for optimizing one-class classifiers. Journal of Machine Learning Research 2, 155–173 (2001)
3. Zhang, H., Huang, W., Huang, Z., Zhang, B.: A kernel autoassociator approach to pattern classification. IEEE Transactions on Systems, Man and Cybernetics - Part B 35(3), 593–606 (2005)
4. Manevitz, L., Yousef, M.: One-class document classification via neural networks. Neurocomputing 70(7-9), 1466–1481 (2007)
5. Wu, S., Banzhaf, W.: Combatting financial fraud: A coevolutionary anomaly detection approach. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 1673–1680 (2008)
6. Markou, M., Singh, S.: Novelty detection: A review – part 1: Statistical approaches. Signal Processing 83, 2481–2497 (2003)
7. Markou, M., Singh, S.: Novelty detection: A review – part 2: Neural network based approaches. Signal Processing 83, 2499–2521 (2003)
8. Curry, R., Heywood, M.: One-class learning with multi-objective Genetic Programming. In: Proceedings of the IEEE Systems, Man and Cybernetics Conference, SMC, pp. 1938–1945 (2007)
9. Koza, J.: Genetic programming: On the programming of computers by means of natural selection. Statistics and Computing 4(2), 87–112 (1994)
10. Curry, R., Lichodzijewski, P., Heywood, M.: Scaling genetic programming to large datasets using hierarchical dynamic subset selection. IEEE Transactions on Systems, Man and Cybernetics - Part B 37(4), 1065–1073 (2007)
11. Kumar, R., Rockett, P.: Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution: A pareto converging genetic algorithm. Evolutionary Computation 10(3), 283–314 (2002)
12. Zitzler, E., Thiele, T.: Multiobjective evolutionary algorithms: A comparitive case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
13. Chiu, S.: 9. In: Fuzzy Information Engineering: A Guided Tour of Applications. John Wiley & Sons, Chichester (1997)
14. Asuncion, A., Newman, D.J.: UCI Repository of Machine Learning Databases. Dept. of Information and Comp. Science. University of California, Irvine (2008), `http://www.ics.uci.edu/~mlearn/mlrepository.html`